IS41070 Project [15 Credits]

IS41070 Machine Learning Foundations - Academic Year 2024/2025
Deadline: 25th of July 2025
Module Coordinator: Arjumand Younus
Teaching Assistant: Zaur Gouliev

Overview:

In this project, you will apply the concepts learned throughout the module to an open-ended machine learning problem of your choice. You will choose your own dataset (e.g., from Kaggle, the UCI Machine Learning Repository, HuggingFace Datasets, or other reputable open sources) and define a task to solve with it. You may select any type of machine learning task that fits your data: binary classification, multi-class classification, regression, or unsupervised clustering. For example, you might decide to detect fake news (binary classification), perform sentiment analysis on movie reviews or social media posts (classification), predict click-through rates for ads (regression), predict customer churn (binary classification), or songs by their audio features to discover genre groupings (clustering). Choose a **problem that interests you and is feasible with the data you have**. This will require critical thinking and you will be asked to explain your dataset in some detail. The goal is to carry out a complete ML project pipeline on this problem, from understanding the data, through preparation and modeling, to evaluating results and reflecting on implications. You are encouraged to **be creative and challenge yourself**. You can use and explore techniques beyond those covered in class if relevant, as long as the core principles from the module are demonstrated. You can find submission documentations at the end of this document + a rubric on Brightspace.
The project is divided into four main parts:

1] Data Understanding
2] Data Preparation & Modeling
3] Evaluation
4] Reflection

Each part is described below along with the requirements. Work through these parts in order, documenting your process and results as explained. By the end, you should have a well-documented **Python Notebook** showcasing your analysis, along with a brief written reflection on your model's social implications, interpretability, and ethical considerations. This can be a separate word file OR preferably in the markdown text at the bottom of your notebook.

The entire project must be **completed individually**. Use of external code or libraries provided in class is allowed with proper attribution (cite or reference any code snippets or ideas you incorporate). **Ensure all code, analyses, and text are your own work** – plagiarism or uncredited copying will result in a zero grade. Please see the instructions for submitting your work at the end of file.

Task 1: Data Understanding

In this first phase, you will **choose your dataset and get to know it**. This involves sourcing the data, formulating your project question, and performing exploratory data analysis (EDA) to understand what you're working with. This is the recommended structure for it.

1] Select and Introduce a Dataset: Choose a dataset that suits a machine learning task of interest to you. The dataset should be publicly available (from Kaggle, UCI, etc.) or otherwise provided with your submission. It is your responsibility to find this dataset and to make sure it will work as intended. Briefly **describe the dataset** in your notebook: What does it contain (instances, features, target variable)? What question or prediction task are you going to tackle with this data? For instance, *"I will use a news articles dataset from Kaggle to build a model that classifies articles as real or fake news."* Provide any relevant context or source information for the data (including a URL or reference to the source). Ensure the dataset is manageable in size and complexity for this project. Do not get a 16GB dataset or something over the top!

2] Load the Data: Import the data into your Python environment (e.g., using pandas to read a CSV or an appropriate loader for the data format). Verify that the data has been loaded correctly by displaying sample rows or summary information.

3] Explore the Data (EDA): Perform an exploratory analysis to understand the structure and contents of the dataset. Your analysis should be tailored to the nature of your data and the chosen task, but here are some general steps to consider:

- ➢ **Data Overview**: Determine the number of observations (rows) and features (columns). Identify which column is the target variable (for supervised tasks) or note that there is no target if you plan an unsupervised task. If the dataset has class labels, check how many classes and how balanced they are (class distribution); if it's a regression, examine the range and distribution of the target values.
- ➢ **Feature Analysis**: For each feature (or a selection of important features), examine its properties. This could include summary statistics (mean, median, etc. for numeric features), distribution plots or histograms, and unique value counts for categorical features. If your data is text, you might look at things like the most frequent words or phrases, average text length, etc. (For example, in a text dataset you could find the most common terms in each category or the length of articles in each class).
- ➢ **Relationships**: If you have a target variable, explore relationships between features and the target. For numeric features, you might look at correlations with the target or create

scatter plots; for categorical features, maybe compare target averages or counts across categories. If performing clustering (no preset target), look for interesting patterns or groupings in the features (you could do a preliminary visualisation like PCA for high-dimensional data to see potential clusters).

- ➢ **Data Quality Check**: Investigate whether the dataset has missing values, inconsistent entries, or obvious outliers. Identify any anomalies or data quality issues. For example, check for null/NA values in each column, and examine any values that seem out of range or erroneous.
- ➢ **Summarise Findings**: Throughout your EDA, use Markdown cells to **comment on your observations**. What insights have you gained about the data? Note any notable patterns (e.g., "Category A news articles tend to be much longer on average than Category B"), any issues you will need to address (e.g., missing values or class imbalance), and any initial thoughts on how these insights might influence your approach in the next steps.

By the end of Task 1, you should have a clear understanding of what your data looks like, what problem you will solve with it, and any challenges you might need to handle. This section sets the stage for your modeling work, so a thorough EDA and understanding is important.

Task 2: Data Preparation & Modeling

In this part, you will **prepare your data for modeling and then develop machine learning models** to address your chosen task. This involves cleaning and preprocessing the data, selecting appropriate features, and training one or more models. You have flexibility in the methods you apply, but you should **justify your choices** at each step. The quality and reasoning of your preprocessing and modeling are as important as the performance of the final model. You need to have at **least 5 machine learning models trained**.

1] Data Splitting: If you are working on a supervised learning task (classification or regression), split your dataset into training, validation, and test sets. Choose an appropriate split (for example, 60/20/20 or 70/15/15 for train/validation/test are common choices, but you can adjust as needed). Typically, the **test set** should be around 15-20% of the data and kept completely untouched until final evaluation (Task 3), and the validation set can be used for model tuning and selection. Clearly **document** the rationale for your split: why did you choose the given proportions? If your dataset is time-based or temporal aspects matter, ensure the split respects chronological order. **Save these subsets** (e.g., as CSV files `train.csv`, `valid.csv`, `test.csv`) for reproducibility.

If you are working on an unsupervised task (clustering) where there is no obvious train/test split, you may not need a traditional split. In that case, you can use the entire data for analysis and later evaluate *clustering quality with appropriate methods. However, if you plan to tune parameters (like the number of clusters) you might consider using an internal cross-validation or a hold-out of data for validation of your clustering approach.*

2] Preprocessing and Feature Engineering: Prepare your data for input into machine learning models. This step will depend on your data type and chosen algorithms:

- ➢ **Cleaning**: Address any issues found in Task 1. For example, handle missing values (through imputation or removal), correct or remove outliers if necessary, and fix inconsistencies. Explain how you deal with these issues and why (e.g., *"I dropped 5 records with missing target values to avoid complications in training,"* or *"I replaced missing age values with the median age since that feature is needed for prediction."*).
- ➢ **Transformation**: Apply any needed transformations to the features. For text data, this might include lowercasing, removing punctuation or stopwords, stemming/lemmatisation, or converting text to numerical representations (see next bullet). For numerical data, you might normalise or standardise features (especially if using algorithms that are sensitive to scale, like KNN or neural networks). For categorical data, consider encoding categories (one-hot encoding, ordinal encoding, etc.) if needed. If you have date/time data, you might extract useful features (like day of week, etc.). **Always provide reasoning** for each preprocessing step: how does it help your model or clean the data?
- ➢ **Feature Encoding/Representation**: Convert your raw features into a form that can be used by ML models. Text must be converted to numeric features (e.g., using techniques like Bag-of-Words, TF-IDF, or word embeddings). Categorical variables may need to be encoded as numeric values. If you have images or other specialised data, ensure they are processed into feature tensors or arrays suitable for modeling. The outcome of this step should be a **prepared training set** (and similarly processed validation/test sets) where all features are numeric and ready for modeling. Make sure that any preprocessing you apply to the training data is **consistently applied** to the validation and test data as well (for example, if you normalise or encode features, do it using the parameters determined from the training data to avoid data leakage).
- ➢ **Feature Selection (if applicable)**: You might perform feature selection or dimensionality reduction if you have many features. This could include removing irrelevant features, using techniques like PCA, or selecting the top features based on correlation or importance. If you do this, explain why and how you decided to select or transform features.

Model Training – Initial Models: Train **at least FIVE different machine learning models** on your training data. Using multiple models will allow you to compare performance.

- **Choice of Algorithms**: You should choose models that are appropriate for your task and that have been covered (at least conceptually) in the module. For example, for classification you might try algorithms such as Logistic Regression, Decision Trees, Naïve Bayes, Support Vector Machine, or Random Forest. For regression, you could use Linear Regression, Decision Tree Regression, or a Regression SVM, etc. If you're doing clustering, you might experiment with algorithms like K-Means, Hierarchical clustering, or DBSCAN. You are not limited to these examples – feel free to use any ML type that makes sense as long as you can explain them. **For each model you choose, justify why it is a sensible choice** for your problem. For instance, *"I chose a Random Forest classifier because my data has a mix of numerical and categorical features and I suspect an ensemble of trees can capture nonlinear relationships without heavy parameter tuning."*

- **Training Process**: Train each model on the training set. Clearly document any important settings or hyperparameters you use (e.g., number of trees in a forest, kernel type for an SVM, etc.). If you need to adjust some parameters to get the model to converge or to handle the data (for example, setting a maximum number of iterations), note that as well. You do **not** necessarily need to perform exhaustive hyperparameter tuning at this stage – a reasonable default setting is fine for initial models, since you will have a chance to improve models later. However, if you do tune any hyperparameters now (using the validation set), make sure to mention what you tried.
- **Initial Results Recording**: While the detailed evaluation will be done in Task 3, you should at least verify that your models have been trained properly. This could mean checking the performance on the training set (e.g., training accuracy or loss) to ensure the model is learning something, and maybe a quick check on the validation set to see if the performance is reasonable. Don't spend too long here, as thorough evaluation is part of Task 3, but do make sure your models are working as expected (no crashes, reasonable accuracy, etc.).

Model Training - Additional Approach (Optional but Encouraged): To push your project further and make it more challenging, consider training or incorporating an additional modeling approach, such as another **advanced technique**:

➢ If your problem lends itself to a specific machine learning task(this is where you can look at other research that has done similar things, e.g., academic literature!), you could try that approach and justify it based on the literature, e.g., research from Younas or Gouliev et al., applied a K-means or SVM to this problem, and therefore we tried it to our problem.

➢ Alternatively, you might try an ensemble or a different approach than your initial models (e.g., if you tried five single models already, you might also try a voting classifier that combines them, or a more complex model like XGBoost).

➢ This step is not required for full credit if your five initial models are well done, but attempting it and documenting the results will make your project stand out as more ambitious. **If you choose to do this**, explain what you did (e.g., "trained a KM model because…") and any additional challenges you faced (such as needing more data preprocessing or tuning). Ensure that you also include this model in your evaluation in the next section.

Note: You could, if you have experience with, apply a deep learning model. This is entirely optional. If you are new to deep learning and not confident, you can skip a deep model to focus on doing the other models well. Again, this is an optional component of the assignment.

Throughout Task 2, **documentation is key**. Use Markdown cells to explain each step of your preprocessing and modeling process, and add code comments to clarify non-obvious code sections. The markers will look for sound reasoning: we want to see **why** you made each choice (e.g., why you encoded features a certain way, why you chose a particular model, etc.). A well-prepared dataset and a well-chosen set of models will set you up for effective evaluation in the next part.

Task 3: Evaluation

In this part, you will **evaluate the performance** of your models from Task 2, compare them, and iterate to improve them. You will use appropriate metrics to assess how well the models solve the task, perform error analysis to gain insights from their mistakes, and potentially tune or improve the models based on those insights. The evaluation should conclude with selecting a "best" model and testing it on a held-out test set (for supervised tasks) or thoroughly analysing cluster quality (for unsupervised tasks). This section is the largest portion of the project, reflecting the importance of not just building models, but critically analysing their behavior and results.

1] Select Evaluation Metrics: Choose a primary evaluation metric (or metrics) that are suitable for your task. Justify **why** this metric is appropriate. For classification tasks, common metrics include accuracy, F1-score, precision/recall, AUC, etc. (you should choose based on what makes sense — for example, F1 might be better than accuracy if you have class imbalance). For regression, you might use mean squared error (MSE), mean absolute error (MAE), or R^2. For clustering, since you don't have ground truth labels (unless you do, in which case it's actually a classification), you could use metrics like silhouette coefficient or Davies-Bouldin index, and/or qualitatively evaluate the clusters. You can also include secondary metrics (for example, in classification always good to look at a confusion matrix or precision and recall for each class, not just overall accuracy).

Explain what a good value for your chosen metric might be in the context of your problem, possibly by mentioning a baseline: e.g., *"A reasonable baseline is the majority-class classifier which yields 60% accuracy, so we aim to beat that significantly,"* or *"Human-level performance on this task is around 95%, which sets an upper bound."* If you found any reference benchmarks for your problem (say from literature or Kaggle), you can mention those for context. This is where you should do some literature review on the topic.

2] Evaluate on Training vs Validation Data: Evaluate each of your models (from Task 2) on both the training set and the validation set (for supervised learning). This means computing the chosen metrics for how the model performs on data it was trained on, and on data it has not seen (validation). Compare these results to assess **overfitting or underfitting**:

- ➢ If a model performs much better on training data than on validation, it may be overfitting (memorising training specifics). If it performs poorly on both, it may be underfitting (not complex enough or not well-tuned for the data).
- ➢ Present the results clearly, for example in a table or as printed output, and **comment on the differences**. For example, *"Model A achieved 95% accuracy on training but only 70% on validation, indicating overfitting. Model B got 85% on training and 80% on validation, which suggests a better generalisation."*
- ➢ For clustering tasks (unsupervised), evaluate the clustering output in a meaningful way. Since there's no "training vs validation" in the same sense, you might instead split data for validating the stability of clusters or just apply the clustering to all data and evaluate the result. You could, for instance, calculate the silhouette score for the clustering, or if you do have some external labels, measure clustering purity. You could also split the data, cluster on one part and see if similar structures appear on the other, to mimic train/validation for clustering.
- ➢ If you trained an advanced model (like a deep learning model or an ensemble in Task 2), include it in this comparison as well. See how it stacks up against your earlier models.

3] Error Analysis: Perform a qualitative error analysis on your models' results. This involves digging deeper into **where and why the models are making mistakes**:

- ➢ For classification, examine examples of **misclassified instances** for each model. Are there particular classes that get confused with each other? Look at a confusion matrix to see which classes have lower performance. Maybe list a few concrete examples (data points) that were predicted incorrectly and try to understand why. Are the inputs noisy or ambiguous? Are there certain patterns the model fails on?
- ➢ For regression, analyse the **residuals or errors**. For instance, plot the predicted vs actual values to see if there are systematic deviations. Identify if certain ranges of the target or certain subsets of data have higher error. You might find, for example, that your model consistently underestimates high values.
- ➢ For clustering, inspect the clusters formed. You could examine a few data points from each cluster to see if they share meaningful traits, or see if any cluster contains a mix of disparate points (indicating the clustering isn't very clean). If you have true labels (like actual genres for songs in a song clustering task), you can see if clusters correspond to those labels or not.

- ➢ Compare errors across models: do different models make mistakes on the same instances or are their errors different? If you have one model that tends to get something right while another gets it wrong, that's interesting to note. **Discuss what you learn** from these errors. This might highlight shortcomings in the data (e.g., insufficient information to distinguish certain classes) or in the model (e.g., a linear model can't capture a nonlinear pattern).

4] Improve the Models: Based on the insights from the initial evaluation and error analysis, **propose at least one improvement** for each model (or focus on the most promising model) and implement it. This could be:

- ➢ Tuning hyperparameters (e.g., adjusting regularisation strength, tree depth, number of clusters *k*, learning rate in a neural network, etc.) to see if performance improves.
- ➢ Try a different preprocessing approach or feature engineering based on what you observed (e.g., if certain text terms confused the model, maybe try removing those or using n-grams; if a certain feature was problematic, try transforming or excluding it).
- ➢ Addressing class imbalance (if relevant) by resampling techniques or class weights.
- ➢ For clustering, maybe trying a different number of clusters or a different clustering algorithm if the first one wasn't ideal.
- ➢ Essentially, treat this as a brief **model tuning/optimisation** step. You don't need to exhaustively try dozens of things, but make a thoughtful change that you expect could boost performance, and then re-evaluate.
- ➢ After making the change, **re-train the model** (or adjust the existing one) and then **evaluate it again on the training and validation sets** with the same metrics. Did your primary metric improve on the validation set? (Be cautious not to overfit to the validation set during this process – keep changes reasonable in scope).
- ➢ Summarise the effect of the change: *"After tuning the max_depth parameter from 5 to 10, the validation accuracy improved by 3%, indicating the model was initially underfitting,"* or *"Using SMOTE to balance the classes slightly improved the recall for the minority class, at the expense of a small drop in overall accuracy."* If an attempted improvement did **not** help, that's also a valuable finding – report it honestly and speculate why it might not have worked.

Cross-Validation (Optional): If you have enough time and your dataset isn't too large, you can strengthen your evaluation by performing **cross-validation**. For example, merge your training and validation sets (now that you've used the validation for initial model selection) and perform k-fold cross-validation with your best model or models. This can give a more robust estimate of performance and use more data for training. If you do this, report the cross-validated performance (e.g., average and standard deviation of the metric across folds) and comment on how consistent the model is across different subsets of data. This step is optional but can be a good final check, especially if you intend to use all data for the final model.

Final Model Selection and Test Set Evaluation: Based on the above evaluations and improvements, **select the best-performing model** (or the model you deem most suitable) for your task. This could be one of the initial models or the improved version. **Justify why you consider it the "best"** – usually this will be because it has the highest validation performance on your primary metric, but you might also consider factors like complexity, training time, or consistency.

- Once you have chosen your final model, **apply it to the test set** (the one you held out in step 1 of this task, which the model has never seen). Make sure you **preprocess the test data in exactly the same way** as the training data (use the same transformations, encoders, etc., that were fitted on training data). Evaluate the model on the test set using your primary metric (and any other relevant metrics) and report the results. This gives an unbiased assessment of how your model might perform on new, unseen data.
- Compare the test results to your validation results. Are they similar? Ideally, your test performance will be close to what you saw on the validation set or cross-validation. If there is a large drop, it could mean some overfitting or that the validation set wasn't representative – discuss what the difference implies.
- **Save your final model** (and any other significant models) to disk, if you haven't already, so that the grader can load it without retraining. For example, you can serialise the model using joblib or pickle for scikit-learn, or `model.save()` for Keras/TensorFlow, etc. Include the saved model file(s) with your submission.

Re-train on Full Data (Optional Extension): After evaluating on the test set, one common practice if you were to deploy a model is to train a final version using **all available data** (training + validation, and possibly also incorporate the test data if you are confident in the evaluation) to maximize the data used for learning. This is somewhat optional in an academic project since using the test data for training means you can't evaluate again on that same test set fairly. However, for completeness, you can do the following:

- Re-train your best model on the combination of the training and validation sets (i.e., the larger dataset without the hold-out test). You should not include the test data here if you still want to report a fair metric, but since you've already evaluated on test in the previous step, you might include test now to see if more data helps – just be clear that any metric computed on data that was previously test is not a completely independent evaluation.
- If you do this re-training, you can then **compare the model's performance** (for example, on the test set if you reuse it, or via cross-validation on train+val) with the earlier results. Did training on more data improve the model's metric? Often, more data can help a model generalise better. Report what you find (e.g., *"By retraining on the full dataset (train+val), the test accuracy increased from 82% to 84%, suggesting the model benefited from the extra data."*).

If the change is negligible, note that as well.

- ➢ This step is mainly to encourage thinking about how you would deploy the best model using all data. It is not required if you are satisfied with your model's performance and analysis up to the prior step.

By the end of Task 3, you should have thoroughly evaluated your models, improved them, and identified a final model with its expected performance. This section should be rich with discussion on how the models compare and what their strengths/weaknesses are. **Clarity of presentation** is important: include tables, plots (such as confusion matrices, ROC curves, error plots) or other visuals as appropriate to support your evaluation, and make sure to explain what each result means in context. This task carries about 40% of the marks, reflecting the importance of sound evaluation in any ML project.

Task 4: Reflection

In the final section of your project, you will step back from the technical details and **reflect on the broader implications and considerations** of your model and the problem you tackled. This written reflection (which can be done in a Markdown cell in the notebook, or a separate report section if preferred) should be a few paragraphs long and cover the following points:

- ➢ **Interpretation of Results**: Discuss what the results of your model mean in practical or social terms. For example, if you built a fake news classifier with 85% accuracy, what does that imply about its reliability? Are 15% of news articles potentially misclassified? How would that impact users or stakeholders who rely on the model's predictions? If you did a customer churn prediction, what do the results say about your ability to identify at-risk customers, and how might a business use that information?
- ➢ **Why the Model Makes Certain Predictions**: Consider **how your model works** and what factors it is using. Identify which features seemed most important in your model's decisions. Many algorithms can provide insight into feature importance (e.g., coefficients in linear models, feature importance in trees, or using SHAP values for any model). Even if you don't compute these exactly, based on your earlier analysis and domain knowledge, **explain why you think the model predicts certain outcomes**. For instance, *"The classifier often confused sports articles with politics when the headline mentioned cities – location names might have led the model to the wrong class."* Or *"Our sentiment model strongly weighs the presence of words like 'great' or 'terrible', which aligns with intuition."*Essentially, demonstrate that you have thought about the model's internal logic and can articulate it in human terms.
- ➢ **Fairness and Ethical Considerations**: Reflect on any potential **ethical issues or biases** in your project. No dataset or model is completely neutral. Ask yourself:
    - ○ Is the data representative of the real-world population or scenario? If not, the

model might be biased. (For example, if your data for a sentiment model came mostly from one demographic or one type of product, the model might not generalise well to others.)

    - Could the model's errors or predictions lead to unfair or harmful outcomes for certain groups of people? (E.g., a loan approval model might inadvertently be biased against a minority group if the training data had historical bias.)
    - If you did a clustering of songs, could there be any subjective or cultural bias in interpreting clusters as genres?
    - Acknowledge these issues. For instance, *"My model for predicting attrition might raise fairness concerns – if certain features like age or gender were used, it could unfairly influence predictions. I ensured to remove explicitly sensitive attributes, but there could still be proxies for those attributes affecting the model."*

- ➢ **Model Interpretability**: Comment on how interpretable or explainable your solution is. Simpler models (like linear models or small decision trees) are usually easy to interpret, whereas complex ones (like large neural networks or ensembles) are more like "black boxes." Discuss the trade-off between model performance and interpretability in your project. If you used a very complex model, did you do anything to make it more interpretable (such as looking at feature importances or example-driven explanations)? If your model were to be deployed, how important would interpretability be for stakeholders' trust?
- ➢ **Limitations and Future Work**: Every project has its limitations. Briefly mention what you see as the main limitations of your current solution. This could be limitations in the data (e.g., not enough data, or not the right features), limitations in the scope (maybe your model only handles a subset of the problem), or performance issues (maybe the accuracy is still not high enough for real-world use). Additionally, reflect on **what you would do next** or differently if you had more time/data. For example, *"To further improve the model, I would try collecting more balanced data, and perhaps explore a larger neural network for better accuracy,"* or *"Given more time, integrating an explainability tool like LIME or SHAP would help interpret the model's decisions."*

The Reflection is worth 10% of the project marks, so give it due attention. This section is your chance to show a deeper understanding of the **context and impact** of machine learning models, beyond just the numbers. A thoughtful reflection can cover societal impacts, ethical model usage, and insights about the reliability and fairness of your model. Remember, there are no right or wrong answers in this part – it's about demonstrating critical thinking. Make sure your writing is clear and concise.

Documentation and Submission Guidelines

In addition to the above tasks, a portion of your grade will implicitly come from **how well you document and present your work**, as well as how reproducible your results are. Below are guidelines to follow to ensure you meet the expected standards of quality and reproducibility:

- ➢ **Jupyter Notebook Usage**: Your entire project should be done in a Python Notebook (`.ipynb` files). Use **Markdown cells** extensively to provide context, explanations, and commentary for each step of your analysis. The notebook should read like a report that interleaves code with narrative. Someone else (or you in the future) should be able to read it and understand the logic without needing you to explain in person. **Please also provide a .html output for it too; see instructions below for that.**
- ➢ **Code Clarity**: Write clean code with **meaningful variable names** and comments. You don't need to comment every single line, but any complex or non-obvious block of code should have an inline comment or accompanying markdown explanation. For example, if you implement an evaluation or a data manipulation, include a comment to say what it's doing. This makes it easier to follow your thinking. Remember, the grader will have no prior knowledge of your dataset or the task you are trying to solve, so clarity is key.
- ➢ **Submission Name Convention: Please name your submission file** your student number, a title of your dataset. E.g., if I did a ML task on Netflix movies, like categorising movies based on genre, I'd call it "18718545_NetflixAnalysis".
- ➢ **Organisation**: Organise your notebook in a logical order (you can use the Task 1, Task 2, etc. structure as section headings inside the notebook for clarity). It often helps to have a short introduction at the top of the notebook (what the project is about) and a conclusion at the end (which could be your Reflection or a summary of results).
- ➢ **Reproducibility**: Your code should be **fully reproducible**. This means someone running your notebook from start to finish should get the same results you reported (assuming the same data and environment). To ensure this:
  - Set a random seed for any random operations (e.g., if you split data randomly or use algorithms with randomness like random forests initialisation, use `numpy.random.seed` or relevant library's seed function).
  - Whenever you do something that involves randomness, mention in your report that you controlled the seed for consistency.
  - **Save your trained models** (as mentioned in Task 3) and any intermediate results if necessary. In your notebook, show how you save the model to a file. Later, when evaluating, you can load from this file to ensure you're using the exact same model you trained (this is important especially if you clear outputs or re-run the notebook – you don't want slight randomness to change the final model).
  - Include a **requirements file** (`requirements.txt`). If you used something special **Results Integrity**: Before submission, **clear all outputs** in the notebook (in Jupyter: `Cell -> All Output -> Clear`) and then run all cells **in order**

(restart the kernel and run all). It is the same process in Google Colab. This guarantees that your notebook runs cleanly from top to bottom and that the outputs (tables, plots, metrics) are up-to-date and consistent with the code. After running everything, you may keep the outputs in the notebook for the grader to see (which is often helpful, so we can read results without rerunning every time), but ensure they are the result of the current code.

➢ **Export to HTML**: Along with the `.ipynb` file, export your completed notebook to an HTML file (`File -> Download as -> HTML`) so that we have a static copy of your results and analysis. The HTML will be used for quick viewing of your report. Make sure the HTML export shows all relevant plots and outputs (if you followed the step above to run all cells, it will). In Google Colab you will have to find a way to download as a HTML; there are various methods so search online.

➢ **Submission Contents**: You will submit your work as a single archive (e.g., a ZIP file). Include:

  - Your Jupyter Notebook file(s) for the project (the main analysis notebook, and if you split tasks into multiple notebooks, include all of them).
  - The HTML export of the notebook(s).
  - All **saved model files** you produced (so we can load your model without retraining, if needed).
  - Your dataset files, *only if* they are small or you have a specific subset. If the dataset is very large or publicly available, you don't need to include the raw data in the ZIP, but include a small sample or at least the instructions to obtain it. (For example, if using a Kaggle dataset, you can mention the Kaggle link. Ideally you will have cleaned your data after exploration, and then saved it, so including a cleaned version of the dataset named appropriately, e.g., "Cleaned_Netflix_Movies_Dataset.csv" would show better care.
  - The `requirements.txt` file for any additional libraries.
  - Your **training, validation and test files** in .csv/xlsx.
  - **Do not include** any unnecessary files (no giant temp files or unrelated data). Keep the submission organised and limited to what's needed to run and evaluate your project. **This will not help in your overall presentation grade** by including huge unnecessary temp files.

➢ **Academic Integrity**: The work should be your own. It's fine (even encouraged) to consult online resources, documentation, or examples for ideas and help, but you **must reference** any significant external code or solutions you use. A simple comment or markdown citation like "// or # Code adapted from [source]" or a link will suffice. Do not copy entire solutions from elsewhere, and **make sure you understand and can explain every part of your code**. We may ask for clarifications or conduct plagiarism checks.

➢

There will be a grading rubric accompanying this assignment on Brightspace. Some general points here that will help you score a high grade:

- Address all required parts of each task (data splitting, multiple models, evaluation steps, etc.) and ensure the code runs without errors.
- The amount of thought and effort put into EDA, justification of choices, evaluation and improvement, and reflection. Trying to go beyond the minimum (such as trying additional stats, presenting insightful plots, or doing thorough error analysis) will be rewarded.
- How well-organised and well-explained the notebook is. This includes readability of code and explanations, use of headings, and overall coherence.
- While we do not purely grade on getting a "high accuracy," your results should demonstrate that you successfully trained models and improved them. We consider whether the performance is reasonable for the task and data, and whether you correctly interpreted those results. Even if your model isn't perfect, a clear understanding of its behavior and limitations can score high.
- The quality of your reflection in Task 4, showing understanding of the model's impact and any ethical issues. You have all by mow conducted AI audits for impact, model performance and governance so this should come straightforward.
- Note that a **high-complexity project** (using a very large dataset or advanced techniques) is not necessarily required for top marks – it's better to have a well-done project on a modest problem than a half-baked attempt on something overly ambitious. However, creativity and ambition (if executed well) will certainly get positive attention.

➢ **Deadlines**: The final submission is due by **25th of July 2025**. Please start early, especially since finding and understanding a new dataset can take time. Late submissions will be subject to the UCD's late penalty policy.

The aim of this project is that you will produce a machine learning project that not only demonstrates your technical skills but also your ability to communicate results and consider the broader context of data and algorithms. We look forward to seeing a variety of interesting projects.

Good luck, and have fun with it!